# **论软件设计模式及其应用**

摘要

  2019年3月，我单位联合某高校研发了《程序在线评测比赛考试系统》。系统以程序代码在线提交自动评测功能为核心，分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等，支持对接教务平台。在项目中我担任系统架构师，负责架构设计工作。

  本文以该系统为例，主要论述了软件设计模式在项目中的具体应用，着重介绍了装饰器、策略、工厂方法三种模式。装饰器模式实现试题在不同场景下的扩展功能，提高功能定制灵活性；策略模式实现评测机不同语言的编译执行方法，降低模块耦合度；工厂方法模式实现数据库不同类型的统一访问，提高系统的可扩展性。这些模式改善了设计质量和开发效率，最终系统顺利上线，获得了用户的一致好评。

正文

  笔者在一个专为高校建设计算机专业智能教学一体化平台的单位任职，过往成果有《计算机组成原理仿真实验系统》等。2019年3月，我单位联合某大学研发了《程序在线评测比赛考试系统》项目（以下简称为“OJ系统”），以取代原有传统的编程上机考试平台。

  系统以程序代码的在线提交自动评测功能为核心，主要分为题库模块、评测机模块、实验作业模块、考试模块、比赛模块、抄袭判定模块、用户管理模块等。题库模块主要负责试题和测试用例的管理，用户根据试题要求编写程序代码提交到系统，系统将测试用例与程序代码发送给评测机模块，由评测机自动编译、执行、判分，并将结果发送给其他相关模块进行统计；实验作业模块用于在线布置作业，从题库中选取试题，设置截止日期等要求；考试模块用于学生在线考试，按教师预先设置的参数自动从题库随机抽题生成试卷，以及向教务平台上传考试成绩；比赛模块主要用于ACM竞赛的培训；抄袭判定模块用于鉴定代码与他人代码雷同率；用户管理模块负责用户信息的管理。在这个项目中，我担任了系统架构师的职务，主要负责系统的架构设计相关工作。

  一般来说，设计模式可分为创建型模式、结构型模式、行为型模式三种。创建型模式抽象了实例化过程，它们帮助一个系统独立于创建、组合和表示它的那些对象，包括工厂方法、抽象工厂、生成器、原型、单例模式等。结构型模式涉及到如何组合类和对象以获得更大的结构，包括适配器、桥接、组成、装饰、外观、享元、代理等。行为模式涉及到算法和对象间职责的分配，不仅描述对象或类的模式，还描述了它们之间的通信模式，包括观察者、策略等。

  在OJ平台的开发中，我们综合使用了多种设计模式。本文着重对装饰器模式、策略模式、工厂方法模式3种设计模式在该项目中的具体应用进行介绍。

1. 装饰器模式

  OJ系统中，全部的试题数据都存放在题库服务中，包括题目介绍、测试用例等公共内容。同一试题在自由练习、实验作业、考试、比赛等不同使用场景下，需要增加不同的扩展功能。例如：实验作业功能需要限定编程语言，并且只允许用户在给定的代码框架中编辑指定的部分。为了解决此问题，我们使用了装饰器模式，给试题对象动态添加职责。在设计装饰器模式中，我们定义了四种角色：被装饰对象的基类、具体被装饰对象、装饰者抽象类、具体装饰者。被装饰对象的基类定义为Problem类，定义一个可动态添加职责的对象接口；具体被装饰对象定义为ConcreteProblem类，继承自Problem类，用以实现具体试题内容；装饰者抽象类定义为Decorator类，维持一个指向Problem实例的引用，并定义一个一致的接口；具体装饰者类根据不同的场景，分别定义为TaskDecorator、TestDecorator、ContestDecorator等类，用以实现不同场景的扩展功能。通过使用装饰器模式，我们可以在任何时候实现新的装饰器，给试题功能添加新的职责，而无需修改原有试题功能，比继承方式拥有更好的灵活性。

2. 策略模式

  根据业务需求，OJ系统的评测机需要同时支持对C语言、C++、Java、PHP、Python等多种常用编程语言程序的评测。不同语言对应的编译与执行方法存在较大差异，例如：C语言需要先调用cl再调用link，生成exe后执行；Java需要先调用jdk的javac编译器，生成class后，再调用jre执行；Python等语言无需编译可直接解释执行。为了解决此问题，我们采用了策略模式将不同语言的编译和执行方法封装起来，并使它们可以相互替换。在设计策略模式中，我们定义了三种角色：环境角色、抽象策略角色、具体策略角色。环境角色持有一个抽象策略角色StrategyCompile接口的引用，并通过StrategyCompile接口，来实现一个具体的编译和执行方法；抽象策略角色定义所有具体策略类所需的统一访问接口；具体策略角色包装了不同语言编译程序的具体调用方法，给每种语言分别定义了CStrategyCompile、JavaStrategyCompile、PythonStrategyCompile等具体策略类。通过使用策略模式，评测机可根据不同的语言类型，选择不同的具体方法来编译执行程序代码，降低了模块的耦合度，提高了软件的可扩展性和可维护性。

3. 工厂方法模式

  因OJ系统采取分布式架构部署，不同的模块根据不同的业务需求，选用了不同种类型的数据库，包括MySQL、Oracle、SQL Server三种。在程序设计过程中如果直接使用每种数据库的原生访问方法，将会使业务逻辑复杂化，并且在开发完成后很难再变更为其他数据库类型。为了解决这一问题，我们使用了工厂方法模式。在设计工厂方法模式中，我们定义了抽象产品、具体产品、抽象工厂、具体工厂四种角色。抽象产品定义了数据库操作DataAccess类，包括数据库连接开关、查询执行、事务等基本公用方法；具体产品继承自抽象产品类，根据数据库类型，建立MySQLDataAccess、OracleDataAccess、SQLServerDataAccess类，分别进行了不同的具体实现；抽象工厂建立一个DataAccessFactory类，定义一个通用的返回数据库操作类的方法；具体工厂继承自抽象工厂，根据不同的数据库类型，分别实现返回不同种数据库具体的操作类。通过使用工厂方法模式，可以有效解决不同数据库类型对软件程序的影响，当需要更换和添加新的数据库类型时，也不需要修改具体的业务逻辑，具有很好的可扩展性。

总结

  通过采用了以上的设计模式，基本达到了预期的效果。系统自2019年10月正式上线已运行一年有余，在学校的日常教学考试和竞赛培训中投入使用，截至目前已有3000以上的学生用户、评测了70000条以上的程序代码，获得了单位同事领导和学校教师们的一致好评。不可避免的，我们在设计过程中，也存在一些问题和不足，不少开发人员在实现过程中有时还是习惯于原有的过程化设计方法，对模式的使用有些抵触。而且，这些设计模式在应用过程中，往往不是单独使用，需要多个模式综合运用。这方面，我们还缺少相关的经验。

  实践证明，OJ系统项目能够顺利上线，并且稳定运行，与系统选用了合适的设计模式密不可分。经过这次OJ平台的开发后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统的设计，使整个系统能够更加好用，更有效地服务于高校师生。